

Лабораторна робота №3

Засоби оптимізації роботи СУБД PostgreSQL

КВ-94 Суховейко Олексій

Завдання

1. Перетворити модуль “Модель” з шаблону MVC лабораторної роботи №2 у вигляд об’єктно-реляційної проекції (ORM).
2. Створити та проаналізувати різні типи індексів у PostgreSQL.
3. Розробити тригер бази даних PostgreSQL.
4. Навести приклади та проаналізувати рівні ізоляції транзакцій у PostgreSQL.

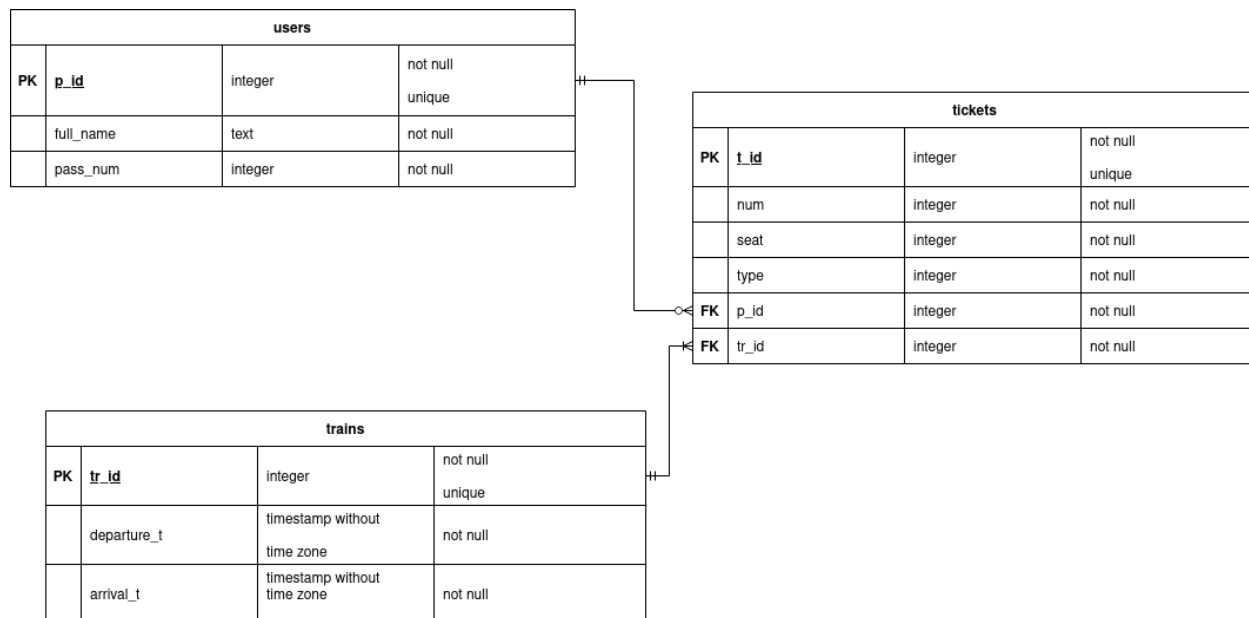
Варіант 23

23	<i>GIN, Hash</i>	<i>before update, delete</i>
----	------------------	------------------------------

URL репозиторію з вихідним кодом

<https://github.com/meltdit/db-labs/tree/master/lab2>

Структура бази даних



Опис структури БД

Таблиця	Атрибут	Опис атрибуту	Тип	Обмеження
users	p_id	Унікальний ідентифікатор	integer	not null unique
	full_name	Повне ім'я користувача	text	not null
	pass_num	Номер паспорту	integer	not null
trains	tr_id	Унікальний ідентифікатор	integer	not null unique
	departure_t	Час вправлення	timestamp without time zone	not null
	arrival_t	Час прибуття	timestamp without time zone	not null
tickets	t_id	Унікальний ідентифікатор	integer	not null unique

	num	Номер вагону	integer	not null
	seat	Номер місця	integer	not null
	type	Тип вагону	text	not null
	p_id	Посилання на користувача	integer	not null
	tr_id	Почилання на потяг	integer	not null

Таблиці у pgadmin4

-- Table: public.users

-- DROP TABLE IF EXISTS public.users;

CREATE TABLE IF NOT EXISTS public.users

```
(
  p_id integer NOT NULL,
  full_name text COLLATE pg_catalog."default" NOT NULL,
  pass_num integer NOT NULL,
  CONSTRAINT users_pkey PRIMARY KEY (p_id)
)
```

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.users
OWNER to postgres;

-- Table: public.trains

-- DROP TABLE IF EXISTS public.trains;

CREATE TABLE IF NOT EXISTS public.trains

```
(
  tr_id integer NOT NULL,
  departure_t timestamp without time zone NOT NULL,
  arrival_t timestamp without time zone NOT NULL,
  CONSTRAINT trains_pkey PRIMARY KEY (tr_id)
)
```

TABLESPACE pg_default;

```
ALTER TABLE IF EXISTS public.trains
  OWNER to postgres;
```

```
-- Table: public.tickets
```

```
-- DROP TABLE IF EXISTS public.tickets;
```

```
CREATE TABLE IF NOT EXISTS public.tickets
```

```
(
  t_id integer NOT NULL,
  num integer NOT NULL,
  seat integer NOT NULL,
  type text COLLATE pg_catalog."default" NOT NULL,
  p_id integer NOT NULL,
  tr_id integer NOT NULL,
  CONSTRAINT tickets_pkey PRIMARY KEY (t_id),
  CONSTRAINT tickets_p_id_fkey FOREIGN KEY (p_id)
    REFERENCES public.users (p_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
  CONSTRAINT tickets_p_id_fkey1 FOREIGN KEY (p_id)
    REFERENCES public.users (p_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
  CONSTRAINT tickets_p_id_fkey2 FOREIGN KEY (p_id)
    REFERENCES public.users (p_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
  CONSTRAINT tickets_tr_id_fkey FOREIGN KEY (tr_id)
    REFERENCES public.trains (tr_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
  CONSTRAINT tickets_tr_id_fkey1 FOREIGN KEY (tr_id)
    REFERENCES public.trains (tr_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID,
  CONSTRAINT tickets_tr_id_fkey2 FOREIGN KEY (tr_id)
    REFERENCES public.trains (tr_id) MATCH SIMPLE
    ON UPDATE NO ACTION
```

```

        ON DELETE NO ACTION
        NOT VALID
    )

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public.tickets
    OWNER to postgres;

```

Класи ORM

```

class users(Base):
    __tablename__ = "users"

    p_id = Column(Integer, primary_key=True)
    full_name = Column(Text)
    pass_num = Column(Integer)

    def __str__(self):
        return "{}, {}, {}".format(self.p_id, self.full_name, self.pass_num)

    def __repr__(self):
        return str(self)

class trains(Base):
    __tablename__ = "trains"

    tr_id = Column(Integer, primary_key=True)
    departure_t = Column(TIMESTAMP)
    arrival_t = Column(TIMESTAMP)

    def __str__(self):
        return "{}, {}, {}".format(self.tr_id, self.departure_t, self.arrival_t)

    def __repr__(self):
        return str(self)

class tickets(Base):
    __tablename__ = "tickets"

    t_id = Column(Integer, primary_key=True)

```

```

num = Column(Integer)
seat = Column(Integer)
type = Column(Text)
p_id = Column(Integer,ForeignKey('users.p_id'))
tr_id = Column(Integer,ForeignKey('trains.tr_id'))

def __str__(self):
    return "{},{},{},{}\n".format(self.t_id,self.num,self.seat,self.type,self.p_id,self.tr_id)

def __repr__(self):
    return str(self)

```

Пункт 1(ORM запити)

Виведення

```

Select table name:

1---users
2---trains
3---tickets

users
Select action(number):

1---Show table items
2---Update table item
3---Insert new table item
4---Delete table item

1
(2,Jane Doe, 21516516)

(1,John Doe, 1337)

(3,c793e74187, 79)

(4,a1527fa03d, 53)

(5,ewasfsdf, 122512)

(333,Kek Kek, 1241241)

Continue to work with Database?(Y/N)

```

Вставка

```
Select table name:
1---users
2---trains
3---tickets

users
Select action(number):
1---Show table items
2---Update table item
3---Insert new table item
4---Delete table item

3
Enter p_id
334
Enter full_name
Joe Black
Enter pass_num
871513
Row was inserted successfully

Continue to work with Database?(Y/N)
Y
```

Початковий стан таблиці

	p_id [PK] integer	full_name text	pass_num integer
1	1	John Doe	1337
2	2	Jane Doe	21516516
3	3	c793e74187	79
4	4	a1527fa03d	53
5	5	ewasfsdf	122512
6	333	Kek Kek	1241241

Після вставки запису





	p_id [PK] integer	full_name text	pass_num integer
1	1	John Doe	1337
2	2	Jane Doe	21516516
3	3	c793e74187	79
4	4	a1527fa03d	53
5	5	ewasfsdf	122512
6	333	Kek Kek	1241241
7	334	Joe Black	871513

Вилучення





```
Select table name:
1---users
2---trains
3---tickets

trains
Select action(number):
1---Show table items
2---Update table item
3---Insert new table item
4---Delete table item
4
Enter tr_id
6
Row was deleted successfully
Continue to work with Database?(Y/N)
Y
```

Початковий стан таблиці

		tr_id [PK] integer 	departure_t timestamp without time zone 	arrival_t timestamp without time zone 
1		1	2001-12-12 00:00:00	2001-12-12 00:00:00
2		2	2001-12-13 00:00:00	2001-12-15 00:00:00
3		3	2001-12-13 00:00:00	2001-12-13 00:00:00
4		4	2001-12-13 00:00:00	2001-12-13 00:00:00
5		5	2003-01-02 00:00:00	2001-12-29 00:00:00
6		6	1234-01-02 00:00:00	1234-01-02 00:00:00

Після вилучення запису

		tr_id [PK] integer 	departure_t timestamp without time zone 	arrival_t timestamp without time zone 
1		1	2001-12-12 00:00:00	2001-12-12 00:00:00
2		2	2001-12-13 00:00:00	2001-12-15 00:00:00
3		3	2001-12-13 00:00:00	2001-12-13 00:00:00
4		4	2001-12-13 00:00:00	2001-12-13 00:00:00
5		5	2003-01-02 00:00:00	2001-12-29 00:00:00









Редагування

```
1---users
2---trains
3---tickets









tickets
Select action(number):
1---Show table items
2---Update table item
3---Insert new table item
4---Delete table item

2
Enter t_id
6
Enter num
133
Enter seat
228
Enter type
second
Enter p_id
334
Enter tr_id
5
Row was updated successfully
Continue to work with Database?(Y/N)
```

Початковий стан таблиці

		 t_id [PK] integer 	num integer 	seat integer 	type text 	p_id integer 	tr_id integer 
1		1	1	1	first	1	1
2		2	1	2	first	1	1
3		3	421421	521512	second	3	4
4		4	1254125	15215	first	4	3
5		5	21421	42154124	first	3	4
6		6	133	228	second	333	5

Після редагування запису

		 t_id [PK] integer 	num integer 	seat integer 	type text 	p_id integer 	tr_id integer 
1		1	1	1	first	1	1
2		2	1	2	first	1	1
3		3	421421	521512	second	3	4
4		4	1254125	15215	first	4	3
5		5	21421	42154124	first	3	4
6		6	133	228	second	334	5

Пункт 2(Індекси gin, hash)

Gin

Створення таблиці:

```
drop table if exists "gin_test";
create table "gin_test"("id" bigserial primary key, "string" text, "gin_vector" tsvector);
insert into "gin_test"("string") select substr(characters, (random() * length(characters) +
1)::integer, 10) from
(values('qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) as
symbols(characters), generate_series(1,1000000) as q;
update "gin_test" set "gin_vector" = to_tsvector("string");
```

Запити для тестування:

```
select count(*) from "gin_test" where ("gin_vector" @@ to_tsquery('bnm'));
select sum("id") from "gin_test" where ("gin_vector" @@ to_tsquery('QWERTYUIOP')) or
("gin_vector" @@ to_tsquery('bnm'));
select min("id"), max("id") from "gin_test" where ("gin_vector" @@ to_tsquery('bnm')) group by
"id" % 2;
```

Запити без індексування:

```
postgres=# select count(*) from "gin_test" where ("gin_vector" @@ to_tsquery('bnm'));
count
-----
19312
(1 row)

Time: 614.007 ms
postgres=# select sum("id") from "gin_test" where ("gin_vector" @@ to_tsquery('QWERTYUIOP')) or ("gin_vector" @@ to_tsquery('bnm'));
sum
-----
24171083391
(1 row)

Time: 1489.679 ms (00:01.490)
postgres=# select min("id"), max("id") from "gin_test" where ("gin_vector" @@ to_tsquery('bnm')) group by "id" % 2;
 min | max
-----+-----
 102 | 999964
    1 | 999969
(2 rows)

Time: 617.664 ms
postgres=#
```

Запити з індексуванням:

```
postgres=# create index "gin_index" on "gin_test" using gin("gin_vector");
CREATE INDEX
Time: 459.084 ms
postgres=# select count(*) from "gin_test" where ("gin_vector" @@ to_tsquery('bnm'));
 count
-----
 19312
(1 row)

Time: 39.162 ms
postgres=# select sum("id") from "gin_test" where ("gin_vector" @@ to_tsquery('QWERTYUIOP')) or
("gin_vector" @@ to_tsquery('bnm'));
      sum
-----
24171083391
(1 row)

Time: 147.338 ms
postgres=# select min("id"), max("id") from "gin_test" where ("gin_vector" @@ to_tsquery('bnm'))
) group by "id" % 2;
 min | max
-----+-----
 102 | 999964
   1 | 999969
(2 rows)

Time: 36.654 ms
postgres=#
```

Пошук з індексацією швидше, тому що кожне значення шуканого ключа зберігається один раз і запит іде не по всій таблиці, а лише по тим даним, що містяться у списку появи цього ключа

Hash

Створення таблиці:

```
create table "hash_test"("id" bigserial primary key, "time" timestamp);
insert into "hash_test"("time") select (timestamp '2021-01-01' + random() * (timestamp
'2020-01-01' - timestamp '2021-01-01')) from
(values('qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM')) as
symbols(characters), generate_series(1,1000000) as q;
```

Запити для тестування:

```
select count(*) from "hash_test" where "time" >= '20010911';
select avg("id") from "hash_test" where "time" >= '20191222' and "time" <= '20211207';
select sum("id"), max("id") from "hash_test" where "time" >= '20180422' and "time" <=
'20210422' group by "id" % 2;
```

Запити без індексування:

```
postgres=# select count(*) from "hash_test" where "time" >= '20010911';
count
-----
1000000
(1 row)

Time: 59.210 ms
postgres=# select avg("id") from "hash_test" where "time" >= '20191222' and "time" <= '20211207';
avg
-----
500000.500000000000000
(1 row)

Time: 101.637 ms
postgres=# select sum("id"), max("id") from "hash_test" where "time" >= '20180422' and "time" <= '20210422' group by "id" % 2;
sum      | max
-----+-----
250000500000 | 1000000
250000000000 | 999999
(2 rows)

Time: 377.160 ms
postgres=#
```

Запити з індексуванням:

```
postgres=# create index "time_hash_index" on "hash_test" using hash("id");
CREATE INDEX
Time: 3655.309 ms (00:03.655)
postgres=# select count(*) from "hash_test" where "time" >= '20010911';
count
-----
1000000
(1 row)

Time: 59.263 ms
postgres=# select avg("id") from "hash_test" where "time" >= '20191222' and "time" <= '20211207';
avg
-----
500000.500000000000000
(1 row)

Time: 103.847 ms
postgres=# select sum("id"), max("id") from "hash_test" where "time" >= '20180422' and "time" <= '20210422' group by "id" % 2;
sum      | max
-----+-----
250000500000 | 1000000
250000000000 | 999999
(2 rows)

Time: 380.876 ms
```

Різниця не значна

Пункт 3(Тригер бази даних)

Таблиці для тестування тригера:

```
DROP TABLE IF EXISTS "trigger_test";
CREATE TABLE "trigger_test"(
    "trigger_testID" bigserial PRIMARY KEY,
    "trigger_testName" text
);
```

```
DROP TABLE IF EXISTS "trigger_test_log";
CREATE TABLE "trigger_test_log"(
    "id" bigserial PRIMARY KEY,
    "trigger_test_log_ID" bigint,
    "trigger_test_log_name" text
);
```

Текст тригера:

```
CREATE OR REPLACE FUNCTION before_update_delete_func() RETURNS TRIGGER as
$trigger$
DECLARE
    CURSOR_LOG CURSOR FOR SELECT * FROM "trigger_test_log";
    row_ "trigger_test_log"%ROWTYPE;

BEGIN
    IF old."trigger_testID" % 2 = 0 THEN
        IF old."trigger_testID" % 3 = 0 THEN
            RAISE NOTICE 'trigger_testID is multiple of 2 and 3';
            FOR row_ IN CURSOR_LOG LOOP
                UPDATE "trigger_test_log" SET "trigger_test_log_name" = '_' ||
row_."trigger_test_log_name" || '_log' WHERE "id" = row_."id";
            END LOOP;
            RETURN OLD;
        ELSE
            RAISE NOTICE 'trigger_testID is even';
            INSERT INTO "trigger_test_log"("trigger_test_log_ID",
"trigger_test_log_name") VALUES (old."trigger_testID", old."trigger_testName");
            UPDATE "trigger_test_log" SET "trigger_test_log_name" = trim(BOTH
'_log' FROM "trigger_test_log_name");
            RETURN NEW;
        END IF;
    ELSE
        RAISE NOTICE 'trigger_testID is multiple of 3';
```

```

        FOR row_ IN CURSOR_LOG LOOP
            UPDATE "trigger_test_log" SET "trigger_test_log_name" = '_' ||
row_."trigger_test_log_name" || '_log' WHERE "id" = row_."id";
        END LOOP;
        RETURN OLD;
    END IF;
END;
$trigger$ LANGUAGE plpgsql;

```

Ініціалізація виконання тригера:

```

CREATE TRIGGER "before_update_delete_trigger"
BEFORE UPDATE OR DELETE ON "trigger_test"
FOR EACH ROW
EXECUTE procedure before_update_delete_func();

```

Дані у таблицях:

```

INSERT INTO "trigger_test"("trigger_testName")
VALUES ('trigger_test1'), ('trigger_test2'), ('trigger_test3'), ('trigger_test4'), ('trigger_test5'),
('trigger_test6'), ('trigger_test7'), ('trigger_test8'), ('trigger_test9'), ('trigger_test10');

```

Запит на оновлення:

```

UPDATE "trigger_test" SET "trigger_testName" = "trigger_testName" || '_log' WHERE
"trigger_testID" % 2 = 0;

```

	trigger_testID [PK] bigint	trigger_testName text
1	1	trigger_test1
2	2	trigger_test2_log
3	3	trigger_test3
4	4	trigger_test4_log
5	5	trigger_test5
6	6	trigger_test6
7	7	trigger_test7
8	8	trigger_test8_log
9	9	trigger_test9
10	10	trigger_test10_log

	id [PK] bigint	trigger_test_log_ID bigint	trigger_test_log_name text
1	1	2	trigger_test2
2	2	4	trigger_test4
3	3	8	trigger_test8
4	4	10	trigger_test10

Запит на видлення:

DELETE FROM "trigger_test" WHERE "trigger_testID" % 3 = 0;

	trigger_testID [PK] bigint	trigger_testName text
1	1	trigger_test1
2	2	trigger_test2
3	4	trigger_test4
4	5	trigger_test5
5	7	trigger_test7
6	8	trigger_test8
7	10	trigger_test10

	id [PK] bigint	trigger_test_log_ID bigint	trigger_test_log_name text

Запити виконані підряд:

	trigger_testID [PK] bigint	trigger_testName text
1	1	trigger_test1
2	2	trigger_test2_log
3	4	trigger_test4_log
4	5	trigger_test5
5	7	trigger_test7
6	8	trigger_test8_log
7	10	trigger_test10_log

	id [PK] bigint	trigger_test_log_ID bigint	trigger_test_log_name text
1	1	2	___trigger_test2_log_log_log
2	2	4	___trigger_test4_log_log_log
3	3	8	___trigger_test8_log_log_log
4	4	10	___trigger_test10_log_log_log